Software toolkit for modeling, simulation and control of soft robots

Eulalie Coevoet, Thor Morales-Bieze, Frederick Largilliere, Zhongkai Zhang, Maxime Thieffry, Mario Sanz-Lopez, Bruno Carrez, Damien Marchal, Olivier Goury, Jeremie Dequidt, et al.

FULL PAPER

Software toolkit for modeling, simulation and control of soft robots

E. Coevoet, T. Morales-Bieze, F. Largilliere, Z. Zhang, M. Thieffry,
M. Sanz-Lopez, B. Carrez, D. Marchal, O. Goury, J. Dequidt, C. Duriez*.

ARTICLE HISTORY
Compiled October 26, 2017

Abstract
The technological differences between traditional robotics and soft robotics have an impact on all of the modeling tools generally in use, including direct kinematics and inverse models, Jacobians, and dynamics. Due to the lack of precise modeling and control methods for soft robots, the promising concepts of using such design for complex applications (medicine, assistance, domestic robotics...) cannot be practically implemented.

This paper presents a first unified software framework dedicated to modeling, simulation and control of soft robots. The framework relies on continuum mechanics for modeling the robotic parts and boundary conditions like actuators or contacts using a unified representation based on Lagrange multipliers. It enables the digital robot to be simulated in its environment using a *direct* model. The model can also be inverted online using an optimization-based method which allows to control the physical robots in the task space. To demonstrate the effectiveness of the approach, we present various soft robots scenarios including ones where the robot is interacting with its environment. The software has been built on top of SOFA, an open-source framework for deformable online simulation and is available at https://project.inria.fr/softrobot/

KEYWORDS
Soft-robot; Simulation; Finite Element Method; Kinematics; Dynamics; Inverse models; 3D

1. Introduction

Soft robotics raises interdisciplinary challenges involving material science, mechanical and electrical engineering, control theory, chemistry, physics, biology, computational mechanics and computer science. While the term *soft* is used, it actually means *non rigid* and is therefore employed for robots whose mechanical functioning relies on using deformable structures in a way similar to the biological world and organic materials. The use of deformable materials makes them very compliant, which provides natural positive key outcomes. Soft robots exhibit new types of functional capabilities that are complementary to traditional robotics. They can improve the safety of access to fragile parts of an environment by applying minimal pressure to its walls. Moreover, their large number of degrees of freedom combined with a redundant actuation can ease the manoeuvring through soft and confined spaces. This is particularly relevant for medical and surgical robotics [1], manipulation of fragile objects, domestic robotics

*Authors are in *Defrost* team: INRIA, CNRS, University of Lille and Ecole Centrale de Lille, France. contact: christian.duriez@inria.fr with safer interactions with humans, arts and entertainment [2].

However, these outcomes often require a complex design. Building robots capable of complex tasks relies on having modeling and simulation tools [3], which is now a standard element in toolkits dedicated to rigid robotics. However, no such tool exists in soft robotics. The main reason is related to the motion of soft robots obtained through deformation of the structure rather than by articulations. Therefore, the behavior of soft robots should be modeled using deformable mechanics. Quoting [1]:

> “There exists well established theories as mechanics of continuous media. In robotics, we need to extract minimal models exploitable for analysis, for control, and to help direct goal-oriented design in particular toward control. In this respect, it will require a big effort to build generic modeling tools suited to soft robotics”.

The presented work is our contribution to this *”big effort”*. The use of continuum mechanics raises several issues. No analytic solution exists in the general case and numerical methods, typically the Finite Element Method (FEM), have to be used. This involves the discretization of the robot geometry which is not trivial (quality of the elements, trade-off between accuracy and computation time. . .). In addition, due to their natural compliance, soft robots are often used in contact with their environment, which increases the complexity of the modeling as well as the computational cost as identified in recent surveys about deformable robots [2], [4], [5], [6].

In this paper, we present a new software framework to model and simulate soft robots and their environment. The framework uses continuum mechanical modeling of soft materials combined with the Finite Element Method (FEM) for their numerical resolution. Boundary conditions are defined using constraints for both contacts and robot actuators. This framework unifies several of our previous works among which: the methodology of the inverse optimization to transfer the motion from task space to motion space presented in [7] with direct simulation [8], dynamic and quasi-static formulations [9]. It also provides contact management for direct simulation as in [10], and for inverse simulation as in [11], and apparent stiffness control of the structure in case of redundant actuation [12].

The framework is implemented as a plugin [13] for SOFA, an open-source toolkit geared towards interactive medical simulation. The motivations to use a medical simulation framework for robotic applications are numerous. Medical simulation and soft robotics make use of strongly deformable materials in complex arrangements. SOFA allows complex simulations and features many deformable models, several spatio-temporal integration schemes and accurate contacts management. It also interfaces many hardware sensors or haptic devices and finally can be run both offline and online. Using this plugin, we simulate soft robots from the state of the art [14] as well as our own prototypes which include robots for grasping, navigating, handling objects or interacting with humans.

This paper is organized as follows: in Section 2, we provide an overview of the modeling and simulation tools for soft robotics. Section 3 contains the theoretical foundation of our framework. Section 5 explains how direct and inverse models are implemented. Section 6 contains implementation aspects related to the SOFA plugin and Section 7 presents examples of soft robots modeled and simulated with our framework.

[1]”First robosoft working paper”, Future Emerging Technology (FET) on Soft-Robot, Technical Report, 2014.

2. Related work

Soft robotics is a very recent and active field where researchers are actively exploring robots designs and their usages. In the following we will focus the state of the art on existing approaches for modeling and simulation of deformable robots.

2.1. Modeling Soft Robots

In soft robotics, softness can be achieved with various approaches from soft materials like silicone [15], [16] or elastomeric polymers [17], micro-structured materials [18] or specifically designed geometrical arrangement of rigid parts as with Tensegrity structures [19]. In addition to the material itself, actuation systems are very diverse with approaches including cables [14], pneumatics [17], [20], shape memory alloys [21] or chemical reaction [22].

Just like in the case of their rigid counterparts, deformable continuum manipulators are the most prominent class of soft robots. They serve as an alternative to the classic industrial rigid manipulators in applications that put the robot in the same workspace as humans. Often bio-inspired [23] [24] [25] [26], these robots are mainly composed by an elongated structure or backbone which is actuated, intrinsically or extrinsically [27], to achieve a certain pose. The compliant nature of this class of manipulators make human-robot interaction very safe; the force in what could be a hazardous collision between the robot and the human gets absorbed by the compliance of the robot. This characteristic makes deformable continuum manipulators particularly suitable for minimally invasive surgery [28] [29] and inspection applications [30], with the trade-off of being inherently difficult to control due to their often non-linear and complex dynamics [31].

A first approach towards the kinematic modeling of continuum manipulators, to relate the configuration of the robot (the shape of its backbone) to the task space and actuation, uses the assumption that the pose of the manipulator under the effects of actuation complies to an arc segment with constant curvature [32]. The relationship between both ends of the backbone is then represented by three discrete transformations. Using the Denavit-Hartenberg approach, one can derive the kinematic model of the backbone. This approach is the most common in the literature related to kinematic modeling of continuum manipulators and has been used to model different types of robots [33] [34] [35]. Nevertheless, this approach is entirely geometric-based and does not capture the mechanics of the material from which the robot is constructed and limits these models in scenarios where the robot is carrying a payload and the effects of gravity are not trivial.

Cosserat rod theory has been used to derived geometrically exact models for continuum manipulators. This model is often used for tendon-based designs in which the deformation of the robot is caused by the forces applied by tendons at specific point along the structure [36]. The model is also used to model manipulator with hybrid actuation tendon and air muscles [37] [38] and concentric tubes designs as well [39]. While these models can account for the non-linear behavior induced by elastic materials, they are limited in terms of the structural geometry of the robot and cannot be used with complex robot shapes, see for example the parallel soft robot in Figure 7.

2.2. Simulating soft robots

The aforementioned model can be used to simulate the behavior of soft robots. In the field of rigid robotics, one can use either dedicated products like WorkspaceLt [40], RoboticSimulation [41], NI-Robotics [42] RoboNaut [43] or SimRobot [44] or general purpose open-source software like Gazebo [3]. The cited tools rely on off-the-shelves simulation kernels such as Open Dynamics Engine [45], Bullet Physics [46], NVidia PhysX [47] or DART [48]. These simulation kernels come from the video-game industry and are often focused on articulated-rigid bodies. They have been successfully used to model and simulate soft robots as in the NASA Tensegrity Robotics Toolkit [19] or in [49] with the use of PhysX to evaluate the candidate solutions of genetic algorithms. The video-game based simulation frameworks are fast and efficient to compute rigid-body simulations as well as some kind of soft bodies. They are also relatively easy to use as required background knowledge in physical modeling is reduced. The counterpart is that very few of them are capable of modeling physically realistic deformable materials.

When a realistic deformable material simulation is needed, tools from the structural and multi-physics analysis field, as Abaqus [50] or ComSol [51], are suitable. They rely on precise modeling formulations of continuous mechanics and some of them are capable to handle multi-physics. The cost for such capabilities is the slow computation speed and the fact that a good understanding of physical modeling is required. The consequence is that they are only usable for offline simulation of soft robots in combination with CAD software while designing the soft robotic parts [20]. Simpler alternatives exist such as Voxelyze [52]. Presented in [53], it simulates soft materials undergoing large deformations and is associated with VoxCAD [54], a GUI simplifying the editing of the robot. Voxelyze relies on voxels to represent the object. It is used in [55] to evaluate through simulation the walking capabilities of soft robots produced by genetic algorithms. In [56], the same authors added interaction between the robot and its environment. Nevertheless, with a voxel simulation, it is not possible to approximate some geometrical shapes without an exaggerated number of voxels which leads to an increased computation time. In addition, with Voxelyze, the mechanical model is using beam theory on the lattice supporting the voxels. Such an approach may not capture the continuous material deformation in a realistic manner.

Research in the field of surgical and biomedical simulation also developed simulation framework [57]. The interesting point of these frameworks is the focus on deformable objects and complex interactions. A tool like SOFA can simulate a large choice of mechanical models: from rigid-bodies or mass-spring to one implementing realistic hyperelastic material with FEM [58]. They can operate on a wide range of geometrical descriptions from 1D (curve) and 2D (surface mesh) to 3D (voxels, multi-resolution octrees [59] or hexahedral and tetrahedral mesh). They are capable of handling collisions and contacts precisely as well as to handle multi-physics behaviors [60], [61]. They are also capable of interacting with sensing hardware (Kinect, OptiTrack, LeapMotion) that are commonly used in robotics as well as with haptic devices [62]. A framework like SOFA can be considered as a *bridge* between video-game and structural analysis approaches and is chosen for this work.

The framework ArtiSynth [63] shares many similarities with the framework we propose: It allows to simulate a mix of rigid and deformable (with FEM) structures, it uses Lagrange Multipliers and implement Hard and Soft constraints... One of the difference is the ability of direct control of a soft-robots using the simulation and more precisely, the inverse models. With our framework, we seek for real-time performance and we have demonstrated that we can directly couple the simulation, in real-time, with a real-device.

In the following of this paper, we will present the plugin we realized for SOFA that is dedicated to soft robotics.

3. Modeling for real-time simulation

The theoretical foundations of our simulation framework for deformable objects are the ones of continuum mechanics for the material modeling, Lagrangian multipliers for constraints solving, and Signorini's law for contacts.

Let us start with the formulation given by the second law of Newton, that models the dynamic behavior of a body as:

$$\mathbb{M}(\mathbf{q})\dot{\mathbf{v}} = \mathbb{P}(t) - \mathbb{F}(\mathbf{q}, \mathbf{v}) + \mathbf{H}^T\lambda \tag{1}$$

where $\mathbf{q} \in \mathbb{R}^n$ is the vector of generalized degrees of freedom (for instance, displacement of the nodes of a mesh), $\mathbb{M}(\mathbf{q}) : \mathbb{R}^n \mapsto \mathcal{M}^{n\times n}$ is the inertia matrix, $\mathbf{v} = \dot{q} \in \mathbb{R}^n$ is the vector of velocity. $\mathbb{F}$ represents internal forces applied to the simulated object depending on the current state and $\mathbb{P}$ gathers known external forces. $\mathbf{H}^T$ is the matrix containing the constraint directions while $\lambda \in \mathbb{R}^n$ is the vector of Lagrange multipliers containing the constraint force intensities. In the following, we will present how these different terms can be computed.

3.1. Mechanical modeling

To compute $\mathbb{F}$, one needs to pick a deformation law. The underlying assumption is that all solids are deformable and the amount of deformation depends only on the external loads. This relationship between the loads and resulting deformations is the constitutive equation. A common equation, the Hooke's law, makes the assumption of linearity of material response to loads. Other laws exists to express nonlinear strain-stress relationship, plastic deformations, brittles or hysteresis behavior. Different laws have different computation costs and one has to carefully choose the law that fits best the needs and computation time constraints. Most of the time, we limit the deformation cases to purely elastic behavior: the robot goes back to its initial shape when the actuation is released and the parameters of the materials are given by the Young's modulus and the Poisson's ratio of the Hooke's law. Different levels of complexity exist in the elastic deformation law which are: *small displacements, large displacements, large deformations* [64] but for most of our robots, we rely on *large displacements* where a non-linear computation is performed to obtain the strain with a linear stress-strain relationship.

3.2. Discretization

Depending on the constitutive equations and the geometrical representation, several possibilities exist in SOFA to model deformable materials. When dealing with 1D structures, one can use beam elements [65] or geometrical curves as in [66]. For 3D structures, there exist mass-spring models, co-rotational FEM [67], embedded deformable solids [59] as well as hyperelastic models to handle *large deformations* [58]. More concretely, each of these models can compute the $\mathbb{F}$ term in Equation 1.

Assuming 1D or 2D geometries, we directly use the provided discretization as a support to solve the constitutive equations. Assuming a 3D arbitrary geometry, the common workflow to build a mechanical model from this geometry is to discretize the domain bounded by the 3D geometry. This is done in SOFA using the open source library CGAL where tetrahedra / hexahedra are generated given a 3D polygonal mesh. In order to have interactive computation times, we adjust CGAL parameters to have the best accuracy given a higher bound of 5000 volume elements. In Figure 1, we show the 3D meshes of some of the robots presented in this work. Once this discretization is performed, the user select a constitutive law and material properties to have a full mechanical model of the soft robot. According to Equation 1, the degrees of freedom are the vertices of the volume elements created.

If the robot is hybrid, SOFA offers a way to combine deformable model and rigid ones. This is done through the *mapping* mechanism [58] which allows to transfer positions, velocities and forces between the deformable model and the rigid components.

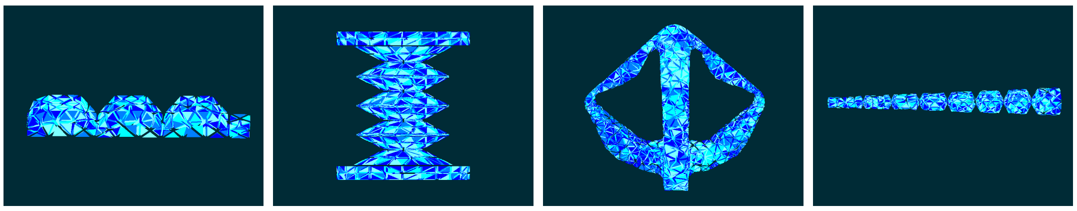


Figure 1. Three dimensional volume mesh of some of the robots presented in the paper. From left to right: one soft finger actuated by one cable (see Flexo 7.2.2), one pneumatic element of Fetch 7.2.1, a parallel like soft robot (Diamond 7.2.3) and a soft trunk actuated by 8 tendons (see 7.2.5).

4. Constraint based definition of boundary conditions

To accurately capture the deformations of soft robots, the FEM integrates over the mesh of the robot, the intrinsic contitutive law of its material. But to create accurate deformation, a careful attention must be paid to the boundary conditions. In this section, we described how we use Lagrange multipliers to model the load applied by the actuators, by the contact with the environment or on the end effector of the robot. The localization of these loads does not necessarily correspond to a node of the FEM mesh.

4.1. Constraint mapping

When defining the geometrical location of constraints, like contact points or the attachment point and passing points of a cable for example, there is not necessarily the nodes of the FEM mesh at this location. To allow these points to be located in the middle of an FEM element, we use a concept of mappings (that is based on the interpolation principle at the basis of the FEM). We connect the constraint points $\mathbf{x}$ to the degrees of freedom $\mathbf{q}$ of the FEM mesh by defining the mapping function $\mathbf{x} = \mathcal{M}(\mathbf{q})$. We can easily derive this mapping so, when computing a derivative of a function $f(\mathbf{x})$ at the constraint level with respect to the degrees of freedom $\frac{\delta f}{\delta \mathbf{q}}$, we can compute it using $\frac{\delta f}{\delta \mathbf{x}} \frac{\delta \mathcal{M}}{\delta \mathbf{q}}$. The computation of the Jacobian of the mappings $\frac{\delta \mathcal{M}}{\delta \mathbf{q}}$ is already implemented in SOFA.

4.2. Actuation constraint

In our framework, we handle the actuation by defining specific constraints with Lagrange multipliers on the boundary conditions of the deformable models. Two types of actuators are considered in this work, cables and pneumatic actuators.

4.2.1. Cable actuation

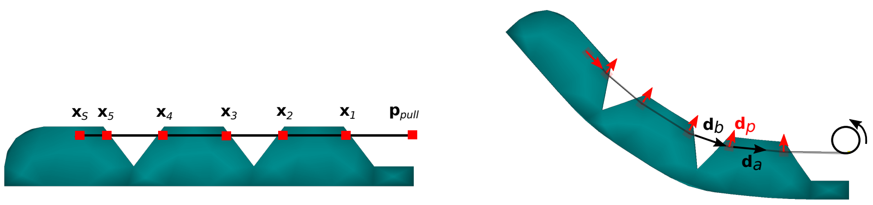


Figure 2. Cable actuation of a soft finger. The cable is passing through the whole finger structure. This cable path enables to control the position of the finger tip.

This actuation is done by placing cables inside the structure of the robot to pull at certain points and create a deformation. To directly link the cable length to the actuator motion, we assume that the cables are not extensible. For each cable, given the pulling point position $\mathbf{p}_{pull}$, we can define the function $\delta_{\mathrm{a}}(\mathbf{x}) : \mathbb{R}^{3n} \rightarrow \mathbb{R}$ that provides its length (which is modified by the actuation). We can constrain the cable length to a maximum and minimum value, $\delta_{\mathrm{a}}(\mathbf{x}) \in [\delta_{min} \ , \ \delta_{max}])$. In the most simple cases, the cable is attached to only one position of the robot and creates a force oriented in the direction of the attachment. In this case, $\delta_{\mathrm{a}}(\mathbf{x}) = \|\mathbf{x}_s - \mathbf{p}_{pull}\|$, with $\mathbf{x}_s$ being the position on the robot where the cable is attached. However, we can use more complex paths for the cable inside the robot. In that case, the length $\delta_{\mathrm{a}}(\mathbf{x}) = \|\mathbf{x}_1 - \mathbf{p}_{pull}\| + \sum_{i=2}^{N} \|\mathbf{x}_{i+1} - \mathbf{x}_i\|$, with $\mathbf{x}_i$ being each position of the model where the cable is passing through (see Figure 2). $\boldsymbol{\lambda}_a$ is the force applied by the cable on the structure.

The corresponding matrix $\mathbf{H}$ is built as follows. At each point x_i, $i \in \{0, 1, .., N\}$, we take the direction of the cable *before* $\mathbf{d}_b$ and *after* $\mathbf{d}_a$ (see Figure 2). To obtain the direction of the constraint that is applied on the point, we use $\mathbf{d}_p = \mathbf{d}_a - \mathbf{d}_b$. Note that the direction of the final point x_s is equal to $\mathbf{d}_a$ as $\mathbf{d}_b$ is not defined. Each constraint directions is stored in the matrix $\mathbf{H}$ (which has the size of the number of nodes) at its corresponding position:

$$\mathbf{H} = \left[\ \ldots \quad \mathbf{d}_p^T \quad \ldots \ \right] \tag{2}$$

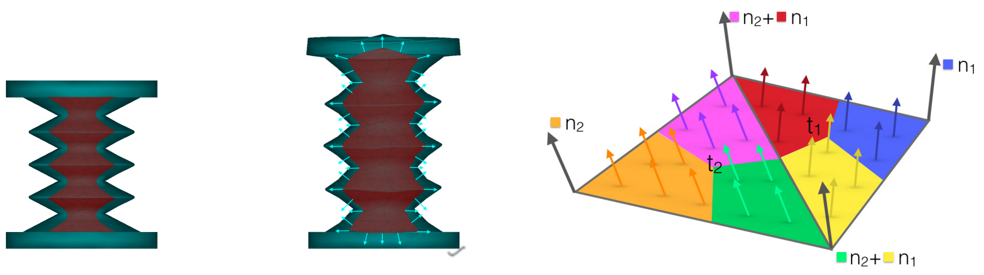


Figure 3. Pneumatic actuation of a soft body with a cavity, at rest (left) and with the cavity inflated by air pressure (middle). The pressure applied on the cavity walls is integrated using the surface of triangles and distributed on the vertices (right).

4.2.2. Pneumatic actuation

This actuation is done by exerting a variation of pressure on the surface of the deformable material. In such a case, $\boldsymbol{\delta}_a(\mathbf{x})$ is a measure of the volume of the cavity and $\boldsymbol{\lambda}_a$ is the uniform pressure on the cavity wall. The corresponding matrix $\mathbf{H}$ is built as follows. For each triangle t of the cavity wall, we compute its area a_t and its normal direction $\mathbf{n}_t$. The multiplication of a_t, $\mathbf{n}_t$ and the pressure $\boldsymbol{\lambda}_a$ gives the vector of force applied by the pneumatic actuation on the triangle t. We distribute this contribution to each of its nodes by dividing the resulting vector by 3 (see illustration in Figure 3). We sum up the results of each triangle in the corresponding column of $\mathbf{H}^T$:

$$\mathbf{f}_i = \sum_{t\in S, i\in t} \frac{a_t}{3}\mathbf{n}_t \lambda_{\mathrm{a}} = (\mathbf{H}^T)_i \lambda_{\mathrm{a}} \tag{3}$$

where $\mathbf{f}_i$ is the force of pressure assigned to the node i and S is the set of the cavity triangles. We can constrain λ_{a} to a maximal or minimal value of pressure. We usually impose pneumatic actuators to only provide positive pressure, $\lambda_{\mathrm{a}} \geq 0$. However, using a vacuum/pressure actuation it is also possible to create both negative and positive pressure.

It is possible to specify the behavior of the actuator either by assigning the value of $\boldsymbol{\lambda}_a$ or by setting the value of $\boldsymbol{\delta}_a(\mathbf{x})$ in the resolution process.

4.3. Contact constraint

In order to add the modeling of the environment, we need to deal with contact mechanics. In the contact case, the function $\boldsymbol{\delta}_c(\mathbf{x})$ measures at the contact point, the shift between the robot and the obstacle. The corresponding matrix $\mathbf{H}$ holds the direction of the contact force, and $\boldsymbol{\lambda}_c$ is the contact response. When a potential contact on the robot has been detected, we need to solve the contact response $\boldsymbol{\lambda}_c$ at the collision point.

For that, we will rely on a formulation of the complementarity problem using Signorini Conditions [68]:

$$0 \leq \boldsymbol{\delta}_c \perp \boldsymbol{\lambda}_c \geq 0 \tag{4}$$

4.4. End effector and task space definition

It is possible to add a load or specify a constraint in the task space. It is particularly useful to simulate a direct or inverse model of the robot (see the following section). A particular point (that we wish to control) is defined on the robot and this point will be considered as the effector. The function $\boldsymbol{\delta}_e(\mathbf{x}) : \mathbb{R}^{3n} \to \mathbb{R}^3$ measures the shift along *x, y* and *z* between this point and the desired position $\mathbf{x}_d$ or trajectory. $\boldsymbol{\delta}_e(\mathbf{x}) = \mathbf{x} - \mathbf{x}_d$ Consequently, Jacobian $\frac{\delta \boldsymbol{\delta}_e}{\delta \mathbf{x}}$ corresponds to identity matrix $\mathbb{I} \in \mathbb{M}_{3,3}$, and matrix $\mathbf{H}$ is easily computed.

Sometimes, it is interesting to define several effector points and/or to control particular directions (*x, y* but not *z* for example). The principle remains the same, it only changes the size of $\boldsymbol{\delta}_e$.

We can also define loads λ_e which are applied on the effector point(s) along the defined direction. Usually these loads are constant and can be easily handled in direct or inverse model using equality constraints. If no load is applied on the effector point(s), then $\lambda_e = 0$.

5. Numerical resolution

In this section, we describe how we integrate in time the equation of the dynamics (Equation 1) and the numerical approaches used to solve the constraints.

5.1. Time integration or quasi-static formulation

We integrate Equation 1 using a time-stepping implicit scheme (backward Euler) to have unconditional stability. Let us consider the time interval $[t_i, t_f]$ whose length is $h = t_f - t_i$:

$$\begin{aligned} \mathbb{M}(\mathbf{v}_f - \mathbf{v}_i) &= h\left(\mathbb{P}(t_f) - \mathbb{F}(\mathbf{q}_f, \mathbf{v}_f)\right) + h\mathbf{H}^T\lambda & (5) \\ \mathbf{q}_f &= \mathbf{q}_i + h\mathbf{v}_f & (6) \end{aligned}$$

The internal forces $\mathbb{F}$ are a nonlinear function of the positions and the velocities. We then apply a Taylor series expansion to $\mathbb{F}$ and make the following first order approximation:

$$\mathbb{F}(\mathbf{q}_f, \mathbf{v}_f)) = \mathbb{F}\ (\mathbf{q}_i + d\mathbf{q}, \mathbf{v}_i + d\mathbf{v}) = \mathbf{f}_i + \frac{\delta \mathbb{F}}{\delta \mathbf{q}} d\mathbf{q} + \frac{\delta \mathbb{F}}{\delta \mathbf{v}} d\mathbf{v} \tag{7}$$

Using $d\mathbf{q} = \mathbf{q}_f - \mathbf{q}_i = h\mathbf{v}_f$ and $d\mathbf{v} = \mathbf{v}_f - \mathbf{v}_i$, we obtain:

$$\underbrace{\left(\mathbb{M} + h\frac{\delta \mathbb{F}}{\delta \mathbf{v}} + h^2 \frac{\delta \mathbb{F}}{\delta \mathbf{q}}\right)}_{\mathbf{A}} \underbrace{d\mathbf{v}}_{\mathbf{dx}} = \underbrace{-h^2 \frac{\delta \mathbb{F}}{\delta \mathbf{q}} \mathbf{v}_i + h\left(\mathbf{p}_f - \mathbf{f}_i\right)}_{\mathbf{b}} + h\mathbf{H}^T\lambda \tag{8}$$

where $\mathbf{p}_f$ is the value of the function $\mathbb{P}$ at time t_f. The only unknown values are the Lagrange multipliers λ ; their computation is detailed in Section 5.2. In the remainder of this section, we will refer to this system using the matrix $\mathbf{A}$ and the vector $\mathbf{b}$.

If the deformable robot is attached to the ground (like a manipulator) and its motion is performed at a low velocity, we can ignore the dynamic part (Equation 1) and use a static formulation:

$$\mathbb{P} - \mathbb{F}\left(\mathbf{q}\right) + \mathbf{H}^T \lambda = 0 \tag{9}$$

Again, the Taylor series expansion (7) can be used to obtain a unique linearization per simulation step:

$$\underbrace{\frac{\delta \mathbb{F}}{\delta \mathbf{q}}}_{\mathbf{A}} \underbrace{d\mathbf{q}}_{\mathbf{dx}} = \underbrace{\mathbb{P} - \mathbf{f}_i}_{\mathbf{b}} + \mathbf{H}^T \lambda \tag{10}$$

We obtain a formulation similar to the dynamic case (Equation 8) with $h = 1$.

5.2. Solving the constraints

From Equation 8 in dynamics or 10 in quasi-statics, the equation has two unknowns: $d\mathbf{x}$ which provides the motion of the degrees of freedom and $\lambda_\mathbf{i}$, which is the intensity of the actuators and contact loads. Consequently, the solving process will be executed in two steps.

The first step consists in obtaining a free configuration $\mathbf{q}_{\text{free}}$ of the robot that is found by solving Equation 10 while considering that there is no actuation and no contact applied to the deformable structure.

$$\begin{aligned} \mathbf{A} d\mathbf{x}^{\text{free}} &= \mathbf{b} \quad &(11)\\ \mathbf{q}_{\text{free}} &= \mathbf{q}_\mathbf{i} + h(\mathbf{v}_\mathbf{i} + d\mathbf{x}_{\text{free}}) \text{ (dynamic)} \quad &(12)\\ \mathbf{q}_{\text{free}} &= \mathbf{q}_\mathbf{i} + d\mathbf{x}_{\text{free}} \text{ (quasi} - \text{static)} \quad &(13)\end{aligned}$$

To solve the linear equation (Equation 11), we use a LDL^T factorization of the matrix $\mathbf{A}$. Given this new *free* position $\mathbf{q}_{\text{free}}$ for all the nodes of the mesh (i.e. position obtained without load on actuation or contact), we can evaluate the values of $\boldsymbol{\delta}_i^{\text{free}} = \boldsymbol{\delta}_i(\mathbf{q}_{\text{free}})$, defined in the previous section.

The second step is based on an optimization process that provides the value of $\boldsymbol{\lambda}$. In the following sections, we will define two cases of use: *direct and inverse modeling.* In both cases, the approach relies on an optimization process and its output is the value of the Lagrange multipliers. The size of matrix $\mathbf{A}$ is often very large so an optimization in the motion space would be computationally very expensive. To perform this optimization in real-time, we propose to project the problem in the constraint space using the Schur complement:

$$\boldsymbol{\delta}_i = \underbrace{h^2 \left[\mathbf{H}_i \mathbf{A}^{-1} \mathbf{H}_j^T\right]}_{\mathbf{W}_{ij}} \boldsymbol{\lambda_j} + \boldsymbol{\delta}_j^{\text{free}} \tag{14}$$

The physical meaning of this Schur complement is central in the method. $\mathbf{W}_{ij}$ provides a measure of the instantaneous mechanical coupling between the boundary conditions i and j, whether they correspond to an effector, an actuator or a contact. In practice, this projection allows to perform the optimization with the smallest possible number of equations.

It should be emphasized that one of the main difficulties is to compute $\mathbf{W}_{ij}$ in a fast manner. No precomputation is possible because the value changes at each iteration. But this type of projection problem is frequent when solving friction contact on deformable objects, thus several strategies are already implemented in SOFA [10], [58].

After solving the optimization process described in the two following subsections (Direct and Inverse modeling), we get the value of $\boldsymbol{\lambda}$, and we can compute the final configuration of the soft robot, at the end of each time step using:

$$d\mathbf{x} = d\mathbf{x}_{\text{free}} + h\mathbf{A}^{-1}\mathbf{H}^T\boldsymbol{\lambda} \tag{15}$$

Which provides the solution to Equation 8 and 10.

5.2.1. *Direct modeling of the robot in its environment*

The inputs are the actuator values (either δ_a or λ_a) and the output is the displacement of the effector. When δ_a is the input, the optimization provides the values of λ_a as output.

As explained above, using the operator $\mathbf{W}_{ea}$, we can get a measure of the mechanical coupling between effector(s) and actuator(s), and with $\mathbf{W}_{aa}$, the coupling between actuators. On a given configuration, $\mathbf{W}_{ea}$ provides a linearized relationship between the variation of displacement $\Delta\boldsymbol{\delta}_e$ created on the end-effector and the variation of the effort $\Delta\boldsymbol{\lambda}_a$ on the actuators. To get a direct kinematic link between actuators and effector point(s), we need to account for the mechanical coupling that can exist between actuators. This coupling is captured by $\mathbf{W}_{aa}$ that can be inverted if actuators are defined on independent degrees of freedoms. Consequently, we can get a kinematic link by rewriting Equation 14:

$$\Delta\boldsymbol{\delta}_e = \mathbf{W}_{ea}\mathbf{W}_{aa}^{-1}\Delta\boldsymbol{\delta}_a \tag{16}$$

This relationship provides (in the most condensed way) the displacement of the effector given the displacements of the actuators. Matrix $\mathbf{W}_{ea}\mathbf{W}_{aa}^{-1}$ is equivalent to a Jacobian matrix for a standard, rigid robot. This corresponds to a local linearization provided by the FEM model on a given configuration and this relationship is only valid for *small variations* of $\Delta\boldsymbol{\delta}_a$, and in contactless cases.

To take into account the contact forces between the robot and its environment and also the self-collisions, the framework allows for contact detection and response. The detection is based on minimal distances computation using an implementation of the algorithm described in [69], adapted to deformable meshes. This algorithm easily manages contact detection between concave meshes, while limiting the number of couples of proximity points, as it selects a couple of points only if they represent a local minimum distance. We can also use an adaptation of the algorithm for self collision, but in practice, we can often predefine the two points of the mesh that will self-collide, and simplify the self-collision detection.

For the collision response, we also use a constraint based approach. Two additional unknowns, δ_c and λ_c are considered in the system. δ_c is a signed distance between the couple of proximity points that have been found by the local minimum distance algorithm. λ_c is the contact force between these two points. δ_c and λ_c follow Signorini's law detailed in Equation 4. In addition, these two values are also linked by the dynamics. In the case of multi-contact, any contact force can modify the distance between the couple of any contact points).

Using the geometric mapping function between the contact distance δ_c and the position of the degrees of freedom $\mathbf{q}$, we can build a Jacobian of contact $\mathbf{H}_c(\mathbf{q})$. λ_c and δ_c are thus linked by Equation 14. Together with the Signorini's law, we obtain a Linear Complementarity Problem (LCP) but if we add some equality constraints to apply the motion created by actuators (we can either set λ_a or $_a$), we obtain a Mixed Complementarity Problem (MCP). Finally, we can augment the problem by defining Coulomb's friction forces, in particular to simulate stick / slip transitions. In such case, we obtain a Non-Linear Complementarity Problem (NLCP). In all cases, we use a block Gauss-Seidel like solver to find a solution (See [70] for details).

This solver provides the values of λ, and is followed by the computation of the final configuration using Equation 15.

5.2.2. Inverse modeling of the robot in its environment

The input is the desired position of the effector and the output is the force λ_a or the motion δ_a that needs to be applied on the actuators in order to minimize the distance with the effector position. λ_a is found by optimization and δ_a can be obtained using Equations 14.

The optimization consists in reducing the norm of $\boldsymbol{\delta}_e$ which actually measures the shift between the end-effector and its desired position. Thus, computing $min(\frac{1}{2}\boldsymbol{\delta}_e^T\boldsymbol{\delta}_e)$ can be done by setting a Quadratic Programming (QP) problem:

$$min\left(\frac{1}{2}\boldsymbol{\lambda_a}^T\mathbf{W}_{ea}^T\mathbf{W}_{ea}\boldsymbol{\lambda_a} + \boldsymbol{\lambda_a}^T\mathbf{W}_{ea}^T\boldsymbol{\delta}_e^{\text{free}}\right) \tag{17}$$

$$\begin{array}{l} \textit{subject to } \text{(course of actuators)}: \\ \delta_{min} \leq \boldsymbol{\delta}_a = \mathbf{W}_{aa}\boldsymbol{\lambda_a} + \boldsymbol{\delta}_a^{\text{free}} \leq \delta_{max} \\ \textit{and } \text{(case of unilateral effort actuation)}: \\ \boldsymbol{\lambda_a} \geq 0 \end{array} \tag{18}$$

The use of a minimization allows us to find a solution even when the desired position is out of the workspace of the robot. In such a case, the algorithm will find the point that minimizes the distance with the desired position while respecting the limits introduced for the stroke of the actuators.

The matrix of the QP, $\mathbf{W}_{ea}^T\mathbf{W}_{ea}$, is symmetric. If the number of actuators is equal or less than the size of the effector space, the matrix is also positive-definite. In such a case, the solution of the minimization is unique.

In the opposite case, i.e when the number of actuators is greater than the degrees of freedom of the effector points, the matrix of the QP is only semi-positive and the solution could be non-unique. In such a case, some QP algorithms are able to find one solution among all possible solutions [71]. In practice, we add to the cost function of the optimization a minimization of the deformation energy in the actuator space: The QP matrix is regularized by adding $\epsilon\mathbf{W}_{aa}$ (with ϵ chosen sufficiently small to keep a good accuracy on the effector motion).

In case of contacts, the response λ_c has to be included in the optimization 17. The corresponding QP problem becomes:

$$min\left(\frac{1}{2}\left[\boldsymbol{\lambda_a}^T\boldsymbol{\lambda_c}^T\right]\left[\mathbf{W}_{ea}^T\mathbf{W}_{ec}^T\right]\left[\begin{array}{c}W_{ea}\\ \mathbf{W}_{ec}\end{array}\right]\left[\begin{array}{c}\lambda_a\\ \boldsymbol{\lambda_c}\end{array}\right]+\left[\boldsymbol{\lambda_a}^T\boldsymbol{\lambda_c}^T\right]\left[\mathbf{W}_{ea}^T\mathbf{W}_{ec}^T\right]\boldsymbol{\delta}_e^{\text{free}}\right) \quad (19)$$

$$\begin{array}{l}subject\ to:\\ \delta_{min}\leq\boldsymbol{\delta}_a=\mathbf{W}_{aa}\boldsymbol{\lambda_a}+\mathbf{W}_{ac}\boldsymbol{\lambda_c}+\boldsymbol{\delta}_a^{\text{free}}\leq\delta_{max}\\ \boldsymbol{\lambda_a}\geq 0\end{array}$$

$$\begin{array}{l}and\ \text{(Signorini Conditions for contacts)}:\\ 0\leq\boldsymbol{\delta}_c=\mathbf{W}_{ca}\boldsymbol{\lambda_a}+\mathbf{W}_{cc}\boldsymbol{\lambda_c}+\boldsymbol{\delta}_c^{\text{free}}\perp\boldsymbol{\lambda}_c\geq 0\end{array} \quad (20)$$

This problem is a QP with Complementarity Constraints (QPCC). Finding the global minimum of such program is difficult to achieve in real-time. We use a specific solver based on the decomposition method we described in [11]. This solver only guarantees a convergence to a local minimum, but gives the real-time performance needed for an interactive control of the soft robots. The idea is that the complementary constraints (20) can be described as a set of linear constraints, given a subset I of $\{1,\ldots,n_c\}$ (with n_c being the number of contacts) giving the elements of λ_c that are forced to be zero. The complementarity constraints (20) then defines 2^{n_c} choices of linear constraints. We rewrite the QPCC as follow:

$$min\left(\frac{1}{2}\left[\boldsymbol{\lambda_a}^T\boldsymbol{\lambda_c}^T\right]\left[\mathbf{W}_{ea}^T\mathbf{W}_{ec}^T\right]\left[\begin{array}{c}W_{ea}\\ \mathbf{W}_{ec}\end{array}\right]\left[\begin{array}{c}\lambda_a\\ \boldsymbol{\lambda_c}\end{array}\right]+\left[\boldsymbol{\lambda_a}^T\boldsymbol{\lambda_c}^T\right]\left[\mathbf{W}_{ea}^T\mathbf{W}_{ec}^T\right]\boldsymbol{\delta}_e^{\text{free}}\right) \quad (21)$$

$$\begin{array}{l}subject\ to\ :\\ \delta_{min}\leq\boldsymbol{\delta}_a=\mathbf{W}_{aa}\boldsymbol{\lambda_a}+\mathbf{W}_{ac}\boldsymbol{\lambda_c}+\boldsymbol{\delta}_a^{\text{free}}\leq\delta_{max}\\ \boldsymbol{\lambda_a}\geq 0\\ and\\ (\boldsymbol{\lambda}_c)_i=0\ ,\ \text{and}\ \ (\boldsymbol{\delta}_c)_i\geq 0\ ,\ \text{for i}\in\text{I}\\ (\boldsymbol{\lambda}_c)_i\geq 0\ ,\ \text{and}\ \ (\boldsymbol{\delta}_c)_i=0\ ,\ \text{for i}\notin\text{I}\end{array}$$

This QP is a piece of (19), we will refer to as QP_{I}. The iterative method starts from an initial feasible set I. We found this initial set by solving the contacts as a Linear Complementarity Constraint (LCP) while considering the actuator force λ_a constant. After solving QP_{I}, we inspect the state of the inequality constraints. If the solution is strictly verifying the inequality constraints, it is a local minimum and the algorithm stops. Otherwise, there are some indices $i_1,\ldots,i_p$ $(1\leq p\leq n_c)$ for which both $(\lambda_c)_i$ and $(\delta_c)_i$ are 0, indicating that the choice I_k might prevent to further decrease the objectives. These indices are kept as candidate for pivot, and I_{k+1} is obtained by adding (resp. removing) a candidate to I_k.

In our algorithm, only one constraint is pivoted at a time. One way to determine which constraint should be the best candidate for pivot is to examine the values of the dual variables. In the implementation, we select the candidate with the greater dual variable.

6. Implementation

In the previous section, we have presented the theoretical foundation of our approach. We will now see more concretely how this translates in the SOFA plugin.

6.1. Concepts of the framework

In a way similar to Gazebo [3], SOFA has a scene-graph based simulation architecture. A scene contains the robot and its environment and is described in XML or with a Python script. SOFA is also a component based architecture. A robot is then an assembly of elementary components: some components are for rendering, others for contact or topology encoding, others for numerical integration or mechanical modeling. For simulation, the most important components are the *Mass* (that computes $\mathbb{M}$), the *Force Fields* ($\mathbb{P}(t)$, $\mathbb{F}$ and $\frac{\delta \mathbb{F}}{\delta \mathbf{q}}$)) and the *MechanicalObject* (which stores the state vectors $\mathbf{q}$, $\mathbf{v}$, $d\mathbf{q}$).

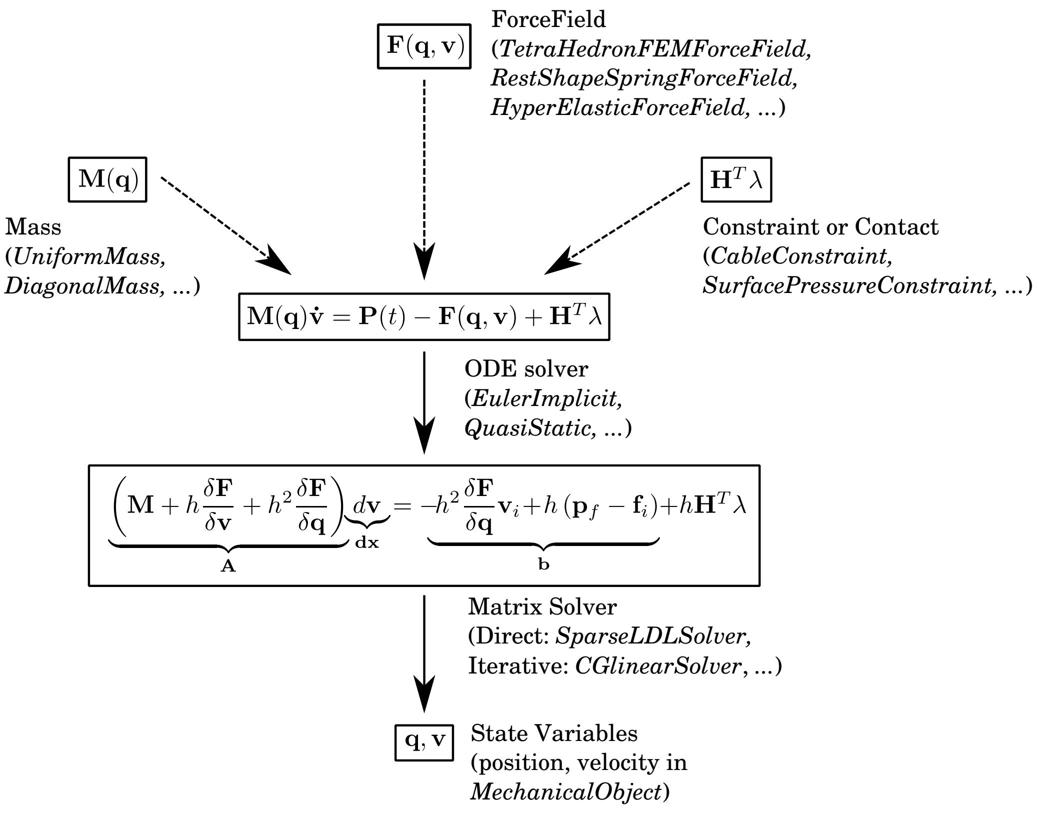

$\mathbf{F}(\mathbf{q},\mathbf{v})$
ForceField
(TetraHedronFEMForceField,
RestShapeSpringForceField,
HyperElasticForceField, ...)
$\mathbf{M}(\mathbf{q})$
Mass
(UniformMass,
DiagonalMass, ...)
$\mathbf{H}^T\lambda$
Constraint or Contact
(CableConstraint,
SurfacePressureConstraint, ...)
$\mathbf{M}(\mathbf{q})\dot{\mathbf{v}} = \mathbf{P}(t) - \mathbf{F}(\mathbf{q},\mathbf{v}) + \mathbf{H}^T\lambda$
ODE solver
(EulerImplicit,
QuasiStatic, ...)
$\underbrace{\left(\mathbf{M} + h\frac{\delta \mathbf{F}}{\delta \mathbf{v}} + h^2\frac{\delta \mathbf{F}}{\delta \mathbf{q}}\right)}_{\mathbf{A}} \underbrace{d\mathbf{v}}_{\mathbf{dx}} = \underbrace{-h^2\frac{\delta \mathbf{F}}{\delta \mathbf{q}}\mathbf{v}_i + h\left(\mathbf{p}_f - \mathbf{f}_i\right)}_{\mathbf{b}} + h\mathbf{H}^T\lambda$
Matrix Solver
(Direct: SparseLDLSolver,
Iterative: CGlinearSolver, ...)
$\mathbf{q},\mathbf{v}$
State Variables
(position, velocity in
MechanicalObject)

Figure 4. Schematic representation of the components in a SOFA scene file.

6.2. Multi-model and mapping

SOFA also introduced multiple model representation. Multi-model means that in a simulated object a property can be represented in multiple ways. A good example is the shape of a robot. A low resolution tetrahedral mesh can be used for FEM while using a high resolution triangular mesh for the rendering and a low resolution triangle mesh for the contact management.

To connect the representations, SOFA introduces *Mappings*. Given the position of the degrees of freedom $\mathbf{q}$ of a representation, one can define a second representation with $\mathbf{x} = \mathcal{M}(\mathbf{q})$, where $\mathcal{M}$ is a possibly non-linear mapping function, as explain in section 4.1.

The strength of *Mappings* is that constraints on a representation can be transferred to a second representation. Using this tool, we can gather actuators, effectors and contacts from different representations to obtain a full system without having to change the implementation of the components. They are defined regardless of the geometrical dimension of the object (1D, 2D, 3D), geometrical representation (voxel, curve, octree, tetrahedral mesh) or mechanical model.

7. Results

The methodology described in this paper allows for the simulation of soft robots in their environment in real-time. Moreover, the inverse model allows online control in the actuator space. The model and control of various robots with different geometric and mechanical characteristics, as well as different actuation schemes, are presented in this section. Table 1 provides a quick overview of the results. The computation timing are based on a modern machine (Intel Core i5-4590 CPU). *This paper is accompanied by a video that allows to have a better understanding of the results.*

Robot	Mat.	Act.	Cnt.	Ctrl.	Num.	Time(ms)
Grasper	S, P	C	Y	D	1422	5
Octoleg	S	C	Y	D	360	1.7
Diamond	S	C	N	I	4884	22
Fetch	S, P	C, P	N	I	12*	7.5
Flexo	P	C	N	I	912	4.5
Stifface	S	C	N	I	2285	30
Trunk	S	C	Y	I	4995	29

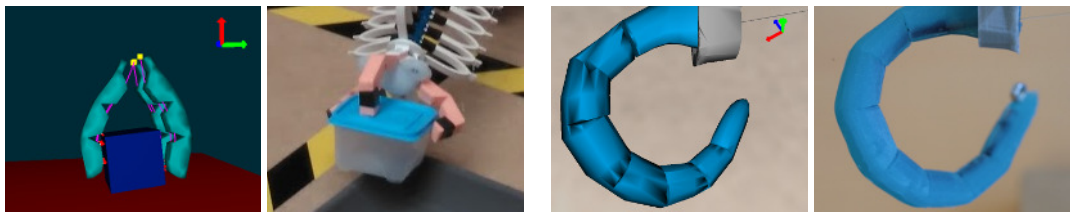
Table 1. The different robots modeled and simulated with their Material (**S**ilicone, 3D Printed **P**lastic), Actuation (**C**able, **P**neumatic), Contacts (**Y**es, **N**o), Control (**D**irect or **I**nverse), Number of Degrees of Freedom and the computation Time (in ms) of one simulation step.
** see paragraph 7.2.1*

7.1. Direct modeling experiments

As explained in 5.2.1, the direct simulation of soft robots takes as inputs the actuator values, while the output is the configuration of the robots given the resultant nodal displacements. In the following, two examples of direct simulation of soft robots are presented.

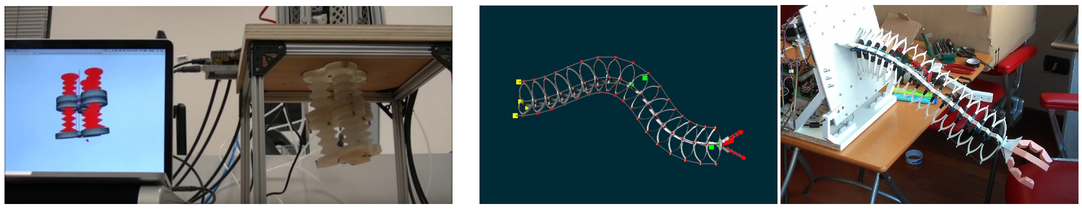
7.1.1. Grasper

In [72], Hassan et Al. present an underactuated grasper design. This grasper is made of silicone and the actuation is done by three cables pulled by a single motor. We successfully modeled this robot and simulated it in real-time (see Figure 5 (a)). The fingers are simulated using corotational FEM and the cube is a rigid body. Corotational FEM


(a) Simulation of the soft grasper described in [72] (left). Grasper of Flexo (right). Comparison between simulated (right) and real.

(b) Simulation (left) of octopus tentacle based on a similar design as described in [72] and real prototype (right).

Figure 5. Simulation and real prototype of different soft graspers.


(a) Control from inverse simulation of two Fetch platforms stacked for the Soft Robotic Toolkit 2015 competition.

(b) Simulation of Flexo (left) and the real robot (right).

Figure 6. Simulation and real prototype of different soft robots.

relies on a tetrahedral mesh[2]. Contact and friction modeling are added to the object to allow prehension. This design evolved into the grasper of Flexo (Subsect. 7.2.2) made of 3D printed plastic.

7.1.2. Octoleg

A close approach of the grasper design described in [72] was used to model an octopus tentacle (Figure 5 (b)). The soft tentacle is simulated and is compared to the physical one entirely made of silicone. The robot is actuated with only one cable going through the whole structure.

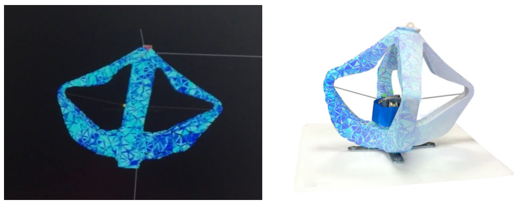
7.2. Inverse modeling experiments

The inverse simulation of soft robots takes as inputs the desired position of the effector and computes the actuation values required to achieved said position (see 5.2.2). This allows for the control of the robots in open loop. In the sequel, some examples of inverse simulation of soft robots are presented.

7.2.1. Fetch: A pneumatic manipulator made of silicone.

The robotic part presented in Figure 6 (a) is a differential pressure platform. Three cavities with a cylindrical accordion shape are inflated to provide an elongation that will extend or tilt the whole element. Several of these platforms can be stacked to

[2]For most of our robots, the volumetric meshes are generated directly in SOFA with the CGAL [73]


(a) Simulated version of the Diamond robot (left) and the physical one (right).

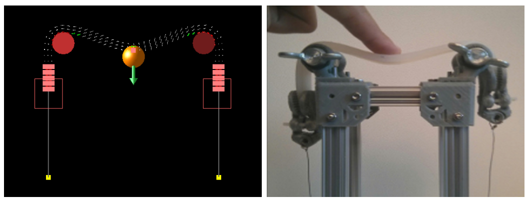

(b) Simulation of the Stifface interface (left) and real prototype (right).

Figure 7. Simulation and real prototype of different soft robots.

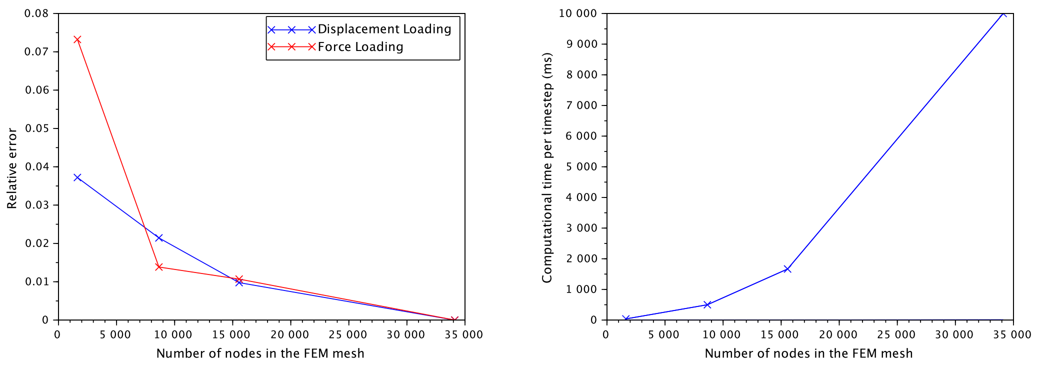


Figure 8. Relative error (left) and computation time (right) of Diamond robot simulation when varying the refinement of the mesh. The relative error is computed for the end-effector position when actuating three of the four cables.

increase the reachable space of the robot. The size of the FEM model of the robot (15456 degrees of freedom) would have prevented from real-time computation. Consequently, we have applied the model reduction method detailed in [9] to obtain a strong reduction of the number of degrees of freedom. The model reduction is based on the structure of the robot (continuum robot with rigid vertebrae). We end up with a model based on the degrees of freedom of the rigid vertebrae (here 12 degrees of freedom). The deformations and the pressure actuation model are *mapped* on these degrees of freedom to accelerate the computation.

7.2.2. Flexo: A manipulator made of deformable material actuated with cables

The robot presented in Figure 6 (b) is composed of several sections, each of which are made of three fork-rib shapes disposed each 120 degrees around the longitudinal direction. Branches have been modeled with beam elements. The robot is actuated with 12 cables; 9 allowing to control the main body motion and three other used to actuate the end grasper. Note that when closing the grasper, a force is exerted to the all structure, and the optimization algorithm finds a new configuration for the actuators to balance this force and solve the desired motion of the robot.

7.2.3. Diamond: A platform made of silicone and actuated through cables.

The Diamond platform (Figure 7 (a)) is made of a single piece of silicone. Four cables, pulling the structure, are connected to servomotors for actuation. For simulation, the silicone is modeled using FEM. The robot can then be controlled, either in the simulation or in the real world, from its endpoint thanks to our inverse simulation method. A stereo-vision system is used to track the position of points of interest in the real robots, which are then compared to those given by the simulation. The maximum positioning error obtained by the inverse model in open loop is 2,9mm, the mean error is 1,4mm. More details are given in [8] and [74]. Using a coarse FEM mesh to achieve the real-time constraint implies some inaccuracies in the simulation of the robot. However, these inaccuracies varies according to the way boundary conditions are enforced on the robot: when displacement are prescribed by the actuators, the size of the mesh has little influence on the prediction of the corresponding displacement of the effector. When forces are prescribed, the refinement of the mesh may have a stronger influence (see bigger error with force loading on the coarsest mesh in Figure 8 (left).). In both cases, we observe a convergence of the model in terms of the refinement of the mesh i.e. finer meshes provide better accuracy. A quantitative comparison shows, in Figure 8 (right), how the simulation time varies with accuracy.

7.2.4. Stiffface: a soft robot acting as a human computer interface

Deformable robotics allows the creation of novel haptics interfaces with soft materials. We use our framework to simulate and control the Stiffface interface (Figure 7 (b)). The device, detailed in [12], is made of silicone and aims to render different apparent stiffness to a user exploring a virtual surface. The silicone is modeled with hyperelastic Neohookean FEM and the cable actuation system is reproduced in the same configuration as in the real device.

7.2.5. Trunk: a cable-driven soft trunk-like robot

With a cable-driven soft trunk-like robot made of silicone, we demonstrate the motion control of a robot interacting with its environment. In the experimental scenario shown in Figure 9 we control the tip of the trunk and the target follows a predefined trajectory.

The robot is actuated with eight cables disposed each 90 degrees around the longitudinal direction. Four cables actuate a first section (from extremity to middle) while the other four go through the entire trunk, allowing it to perform a S-shape. In practice, to avoid friction between cables and silicone, we place flexible tubes along the

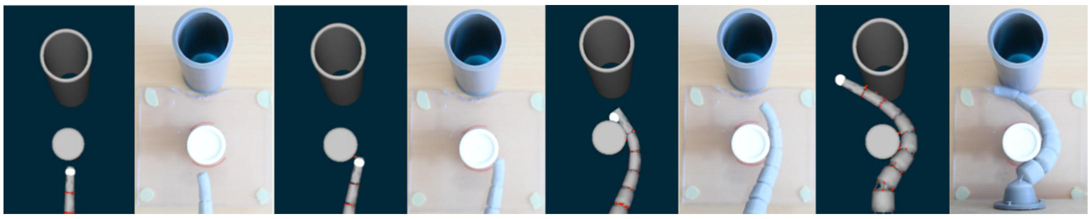


Figure 9. Simulation and real prototype of a cable-driven soft trunk-like robot interacting with its environment. By controlling the motion of the trunk tip using our method, the soft robot was successfully inserted between the two cylinders.

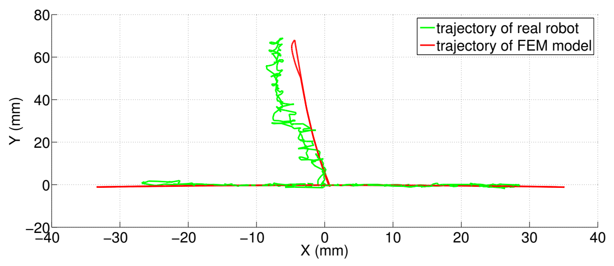


Figure 10. 2D trajectory of the trunk end-effector.

cables path inside the silicone. It allows the cables to slip with low friction. In the simulation, the additional rigidity created by these tubes are modeled using stiff springs. The real trunk is attached to a platform that moves forward and backward along its longitudinal direction. This actuation is also modeled in the simulation. In this way, we were able to interactively drive the trunk end-effector between two cylinders using our optimization algorithm (see Figure 9). In Figure 10, we show a 2D trajectory of the trunk end-effector for both real robot and simulation model (without obstacle). The average error is 4.72mm.

8. Discussion

8.1. Validation

As every simulation based approach, careful validation steps have to be taken for each robot modeled and simulated. The behavior of any simulated robot is sensitive to many properties such as mechanical parameters, mesh discretization, boundary conditions. . . Since our actual robots are proofs of concepts, they do not meet industry criteria of quality control. For instance, while the silicon is being casted, some air bubbles in the robot's body often appear. The structure is not completely homogeneous from a mechanical standpoint. Thus, it theoretically increases possible differences between an ideal, real-time compatible simulated robot and the ones that were built. In order to estimate the modeling and numerical errors that are aggregated in the simulated model, our approach consists in using test-benches composed of tracking devices. These devices (usually IR camera and markers) track the behavior of the real robots moving in all their working space and are compared with their digital counterparts. For instance, the parallel soft robot (called "diamond robot" in Figure 7(a)) has been validated using this methodology and several implementation details can be found in [8] [74]. However, this work has not yet been conducted systematically.

8.2. Performances and accuracy aspects

Several aspects may have an impact on the performance of the simulation. Most significant being the spatial discretization (number of elements) as well as the materials's constitutive equations. As our framework is usable either for interactive simulation or to control real soft robots special care has to be taken depending on the final use of the simulation. Depending on the application, the level of accuracy that is needed varies.

In the Flexo manipulator case (Subsect. 7.2.2) the user is in the control loop. A direct control would be untractable due to the large number of actuators but the inverse simulation is there to help the operator. As long as the general effect of the actuators on the motion of the arm is predicted correctly, small errors in displacement can be tolerable since the user will adapt his movement to succeed in reaching his goal.

In the case of the soft trunk-like robot (Subsect. 7.2.5), which adapt its position according to the contacts it undergoes with the environment, the accuracy level needed in displacement is much higher. Indeed, in this configuration, large errors in displacement can lead to poorly detect contacts and predict dramatically the correct motion of the robot.

8.3. Control laws

The results shown in section 7.2 are open-loop experiments. There is no feedback signal from the real soft robots to compute the control input. This makes the robots sensitive to external disturbances and is known to have a reduced accuracy.

These problems could be solved using a feedback controller. We have started to experience closed-loop approaches within our framework.

A first attempt is to introduce a visual servoing control method. In this approach, the robot is simulated in real-time and an observer make sure that the configurations of both the real robot and its simulation model stays very close. This method allows positional control of the robots as reported in [75]. The drawback of this method is that it is based on a quasi-static model of the robot while neglecting the dynamic part of the model. This affects the performances of the controller. To improve them, we are now considering the use of dynamic models in our control laws. We are thus considering several options. One is to rely on a linearization of the dynamic Equation 1 which gives a large-scale linear state-space model of the soft robot. Model order reduction then transforms the large state-space into one with a size tractable by automatic control methods, such as pole placement [76]. However the state of these works is very preliminary and not yet compatible with inverse modeling strategies detailed above.

As any control method based on a model, the approaches presented in this paper strongly rely on the quality of the underlying modeling. Taking this model uncertainties into account is important and non-trivial but needed to prove the closed-loop stability. A robust control law should be designed to handle both simulation and model order reduction uncertainties.

9. Conclusions

This paper presents the mathematical basis as well as a software framework that targets the design, simulation and control of soft robots. This framework relies on a FEM approach for to handle the mechanical deformations of the robot. Thanks to a set of Lagrange Multipliers defined on the boundary conditions, actuators, effectors and contacts are modeled accurately. These models allow the *direct* simulation of the robot and its interaction with its environment. Moreover, the mechanical representation can be used as an inverse problem optimization that automatically computes the actuation to obtain control in the task space, even in situation of contact with the environment.

The capabilities of this framework are illustrated with several experiments showing that a reasonable accuracy between simulated and real soft robots can be obtained. Matching real-time performance was possible on both the direct and inverse problems by using relatively coarse finite element meshes. If the performance allows to obtain, in average real-time computation, the framework is not implemented in a way that guarantees real-time at each step. We envision to implement such feature, in particular for soft robot control applications.

The modularity of the framework encourages many extensions. For instance, future works may include adding more complex mechanical laws, adding robust control laws or designing complex and dynamic environments. This will increase the computational footprint of the simulation whereas the short computation time needs to be maintained in order to do online control of the robot. Therefore, advanced numerical methods such as reduced-order modeling or dedicated solvers should be considered to achieve sufficient accuracy without increasing the computation cost of the simulation.

Acknowledgment

This project has been financed by the French National Agency for Research (ANR), and by the European Union through the European Regional Development Fund (ERDF).

Note on contributors

Eulalie Coevoet is a Ph.D candidate at INRIA Lille, France. She received the Bachelor's degree in Mathematics from the University Claude Bernard of Lyon, France, in 2011, and the Master's degree in Advanced Scientific Computing from the University of Lille, France, in 2013. From 2013 to 2016, she was a research engineer at INRIA Lille. Her research interests include computer simulation and soft robotics.

Thor Morales-Bieze is a PhD student at Inria Lille, France. He received the Engineering degree from the Institute of Technology Morelia, Mexico, in 2008, and the Master's degree in Control Systems from the University of Michoacan, Mexico, in 2011. His research interests include bio-inspired robotics and soft robotics modeling and control.

Frederick Largilliere is a PhD candidate at the University of Lille, France. He received a Master/French-engineering degree in Mechatronics in 2012 from the ENSIAME, french institute of technology in Valenciennes. His research interests range from designing to modeling, simulating and controlling soft robotic systems for interaction.

Zhongkai Zhang is currently a PhD student in INRIA Lille, France. He received a bachelor degree in Mechanical Engineering and a master degree in Engineering Mechanics. His research interests focus on motion control and force sensing for soft robots.

Maxime Thieffry is a Ph.D student in Control theory at the university of Valenciennes, France. He received a master's degree in automatic control at the university of Lille, France. In addition to control theory, he is studying model order reduction and soft robotics.

Mario Sanz-Lopez received a master degree in telecommunications and electronic systems engineering from the University of Alcala de Henares in Spain. Since 2012 he works as hardware engineer for Inria in Lille. His first works focused on prototype design of new sensors and validation protocols concerning medical simulation and human interfaces for virtual reality environments, moving towards soft robots design and their industrial applications.

Bruno Carrez has been a software engineer for 20 years. After a mechanics engineer degree and 12 years of experience in the video game industry, where he specialized himself in sound engines and physics simulations, he joined Inria in 2012 to work on SOFA, the physics simulation engine used by DEFROST team. His interests include programming good practices, software optimization, 3D printing and everything that has a link with computing in general.

Damien Marchal completed a Ph.D. thesis in Computer Science at the University of Lille 1 in 2006. He then worked for the University of Amsterdam on high performance computing applied to VLBI Radio-Astronomy. Since 2009 he is Research Engineer at CNRS . His interests includes, high performance computing, optimization and compilation techniques as well as human machine interaction applied to geometrical modeling.

Olivier Goury received a PhD in Computational Mechanics from the University of Cardiff in 2015. He was then a postdoctoral fellow at the French National Institute of Health and Medical Research (Inserm), working on the simulation of surgery for cochlear implant insertion. Since the end of 2016, he is a research scientist at Inria, focusing on model order reduction methods for soft robotics.

Jeremie Dequidt completed a Ph.D. thesis in Computer Science at the University of Lille 1 in 2005. As a post-doctoral fellow, he joined the SimGroup at CIMIT (Boston, MA) and then INRIA Alcove team, working on interventional radiology simulations. Since september 2008, he is an Assistant Professor in Computer Science at University of Lille 1. He also is a member of the Inria research-team Defrost. His research interests focus on collision detection and response, mechanical/geometric/adaptive modeling and experimental validations.

Christian Duriez received a PhD degree in robotics from University of Evry, France, in collaboration with CEA/Robotics and Interactive Systems Technologies, and followed by a postdoctoral position at the CIMIT SimGroup in Boston. He arrived at INRIA in 2006 team to work on interactive simulation of deformable objects, haptic rendering and surgical simulation. He is now the head of DEFROST team focused on new methods and software for soft robot models and control, fast Finite Element Methods, simulation of contact response and other complex mechanical interactions and new devices and algorithms for haptics...

References

[1] M. Cianchetti, T. Ranzani, G. Gerboni, T. Nanayakkara, K. Althoefer, P. Dasgupta, and A. Menciassi, "Soft robotics technologies to address shortcomings in today's minimally invasive surgery: The stiff-flop approach," *Soft robotics*, vol. 1, no. 2, 2014.

[2] H. Lipson, "Challenges and opportunities for design, simulation, and fabrication of soft robots," *Journal of soft robotics*, vol. 1, no. 1, 2014.

[3] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proceeding of the conference on Intelligent Robots and Systems. (IROS).*, vol. 3, 2004.

[4] D. Trivedi, C. D. Rahn, W. M. Kier, and I. D. Walker, "Soft robotics: Biological inspiration, state of the art, and future research," *Applied bionics and biomechanics*, vol. 5, no. 3, 2008.

[5] S. Kim, C. Laschi, and B. Trimmer, "Soft robotics: A bioinspired evolution in robotics," *Trends in biotechnology*, vol. 31, no. 5, 2013.

[6] D. Rus and M. T. Tolley, "Design, fabrication and control of soft robots," *Nature*, vol. 521, no. 7553, 2015.

[7] E. Coevoet, N. Reynaert, E. Lartigau, L. Schiappacasse, J. Dequidt, and C. Duriez, "Introducing interactive inverse FEM simulation and its application for adaptive radio-

therapy," in *Proceeding of the Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2014.

[8] C. Duriez, "Control of elastic soft robots based on real-time finite element method," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[9] J. Bosman, T. M. Bieze, O. Lakhal, M. Sanz, R. Merzouki, and C. Duriez, "Domain decomposition approach for fem quasistatic modeling and control of continuum robots with rigid vertebras," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[10] H. Courtecuisse, J. Allard, P. Kerfriden, S. P. Bordas, S. Cotin, and C. Duriez, "Real-time simulation of contact and cutting of heterogeneous soft-tissues," *Medical image analysis*, vol. 18, no. 2, 2014.

[11] E. Coevoet, A. Escande, and C. Duriez, "Optimization-based inverse model of soft robots with contact handling," *IEEE Robotics and Automation Letters (Proceedings ICRA)*, vol. 2, no. 3, 2017.

[12] F. Largilliere, E. Coevoet, M. Sanz-Lopez, L. Grisoni, and C. Duriez, "Stiffness rendering on soft tangible devices controlled through inverse fem simulation," in *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[13] *SoftRobots Plugin for SOFA*. [Online]. Available: https://project.inria.fr/softrobot.

[14] T. Hassan, M. Manti, G. Passetti, N. d'Elia, M. Cianchetti, and C. Laschi, "Design and development of a bio-inspired, under-actuated soft gripper," in *Proceedings of the Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2015.

[15] M. T. Tolley, R. F. Shepherd, B. Mosadegh, K. C. Galloway, M. Wehner, M. Karpelson, R. J. Wood, and G. M. Whitesides, "A resilient, untethered soft robot," *Soft robotics*, vol. 1, no. 1, 2014.

[16] R. V. Martinez, J. L. Branch, C. R. Fish, L. Jin, R. F. Shepherd, R. M. D. Nunes, Z. Suo, and G. M. Whitesides, "Robotic tentacles with three-dimensional mobility based on flexible elastomers," *Advanced materials*, vol. 25, no. 2, 2013.

[17] R. F. Shepherd, F. Ilievski, W. Choi, S. A. Morin, A. A. Stokes, A. D. Mazzeo, X. Chen, M. Wang, and G. M. Whitesides, "Multigait soft robot," *Proceedings of the national academy of sciences*, vol. 108, no. 51, 2011.

[18] C. Schumacher, B. Bickel, J. Rys, S. Marschner, C. Daraio, and M. Gross, "Microstructures to control elasticity in 3d printing," *ACM Trans. Graph.*, vol. 34, no. 4, 2015.

[19] K. Caluwaerts, J. Despraz, A. Işçen, A. P. Sabelhaus, J. Bruce, B. Schrauwen, and V. SunSpiral, "Design and control of compliant tensegrity robots through simulation and hardware validation," *Journal of the royal society interface*, vol. 11, no. 98, 2014.

[20] R. V. Martinez, C. R. Fish, X. Chen, and G. M. Whitesides, "Elastomeric origami: Programmable paper-elastomer composites as pneumatic actuators," *Advanced functional materials*, vol. 22, no. 7, 2012.

[21] A. Villanueva, C. Smith, and S. Priya, "A biomimetic robotic jellyfish (robojelly) actuated by shape memory alloy composite actuators," *Bioinspiration & biomimetics*, vol. 6, no. 3, 2011.

[22] B. Mosadegh, P. Polygerinos, C. Keplinger, S. Wennstedt, R. F. Shepherd, U. Gupta, J. Shim, K. Bertoldi, C. J. Walsh, and G. M. Whitesides, "Pneumatic networks for soft robotics that actuate rapidly," *Advanced functional materials*, vol. 24, 2014.

[23] S. Neppalli, B. Jones, W. McMahan, V. Chitrakaran, I. Walker, M. Pritts, M. Csencsits, C. Rahn, and M. Grissom, "Octarm - a soft robotic manipulator," in *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 2007.

[24] W. McMahan, B. A. Jones, and I. D. Walker, "Design and implementation of a multi-section continuum robot: Air-octor," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005.

[25] T. Zheng, Y. Yang, D. T. Branson, R. Kang, E. Guglielmino, M. Cianchetti, D. G. Caldwell, and G. Yang, "Control design of shape memory alloy based multi-arm contin-

uum robot inspired by octopus," in *9th IEEE Conference on Industrial Electronics and Applications*, 2014.

[26] C. Laschi, M. Cianchetti, B. Mazzolai, L. Margheri, M. Follador, and P. Dario, "Soft robot arm inspired by the octopus," *Advanced robotics*, vol. 26, no. 7, 2012.

[27] I. D. Walker, "Continuous backbone "continuum" robot manipulators," *Isrn robotics*, vol. 2013, 2013.

[28] T. Kato, I. Okumura, S. E. Song, A. J. Golby, and N. Hata, "Tendon-driven continuum robot for endoscopic surgery: Preclinical development and validation of a tension propagation model," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 5, 2015.

[29] N. Simaan, R. Taylor, and P. Flint, "A dexterous system for laryngeal surgery," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 1, 2004.

[30] J. S. Mehling, M. A. Diftler, M. Chu, and M. Valvo, "A minimally invasive tendril robot for on-space inspection," in *The First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics, 2006. BioRob 2006.*, 2006.

[31] H. Hauser, A. J. Ijspeert, R. M. Füchslin, R. Pfeifer, and W. Maass, "Towards a theoretical foundation for morphological computation with compliant bodies," *Biological cybernetics*, vol. 105, no. 5, 2011.

[32] M. W. Hannan and I. D. Walker, "Analysis and experiments with an elephant's trunk robot," *Advanced robotics*, vol. 15, no. 8, 2001.

[33] ——, "Kinematics and the implementation of an elephant's trunk manipulator and other continuum style robots," *Journal of field robotics*, vol. 20, no. 2, 2003.

[34] Q. Zhao and F. Gao, "Design and analysis of a kind of biomimetic continuum robot," in *Robotics and Biomimetics (ROBIO), 2010 IEEE International Conference on*, 2010.

[35] C. Escande, P. M. Pathak, R. Merzouki, and V. Coelen, "Modelling of multisection bionic manipulator: Application to robotinoxt," in *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, 2011.

[36] D. C. Rucker and R. J. Webster III, "Statics and dynamics of continuum robots with general tendon routing and external loading," *IEEE Transactions on Robotics*, vol. 27, no. 6, 2011.

[37] D. Trivedi, A. Lotfi, and C. D. Rahn, "Geometrically exact dynamic models for soft robotic manipulators," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, 2007.

[38] ——, "Geometrically exact models for soft robotic manipulators," *IEEE Transactions on Robotics*, vol. 24, no. 4, 2008.

[39] D. C. Rucker, B. A. Jones, and R. J. Webster III, "A geometrically exact model for externally loaded concentric-tube continuum robots," *IEEE Transactions on Robotics*, vol. 26, no. 5, 2010.

[40] *Workspacelt*. [Online]. Available: http://www.workspacelt.com.

[41] *Roboticsimulation*. [Online]. Available: https://robologix.com.

[42] *Ni-robotics*. [Online]. Available: http://www.ni.com.

[43] *Robonaut*. [Online]. Available: http://robonaut.jsc.nasa.gov.

[44] *Simrobot*. [Online]. Available: http://www.informatik.uni-bremen.de.

[45] *Open dynamics engine*. [Online]. Available: http://ode.org.

[46] *Bullet physics*. [Online]. Available: http://bulletphysics.org.

[47] *Nvidia physx*. [Online]. Available: https://developer.nvidia.com/physx-sdk.

[48] *Dart*. [Online]. Available: http://dartsim.github.io.

[49] J. Rieffel, F. Saunders, S. Nadimpalli, H. Zhou, S. Hassoun, J. Rife, and B. Trimmer, "Evolving soft robotic locomotion in physx," in *Proceedings of the 11th annual conference companion on genetic and evolutionary computation conference: Late breaking papers*, 2009.

[50] *Abaqus*. [Online]. Available: http://www.3ds.com/products-services/simulia/products/abaqus.

[51] *Comsol*. [Online]. Available: https://www.comsol.fr.

[52] *Voxelyze*. [Online]. Available: https://github.com/jonhiller/Voxelyze.

[53] J. Hiller and H. Lipson, "Dynamic simulation of soft multimaterial 3D-printed objects," *Soft robotics*, vol. 1, no. 1, 2014.

[54] *Voxcad*. [Online]. Available: https://sourceforge.net/projects/voxcad.

[55] N. Cheney, R. MacCurdy, J. Clune, and H. Lipson, "Unshackling evolution: Evolving soft robots with multiple materials and a powerful generative encoding," *Sigevolution*, vol. 7, no. 1, 2014.

[56] N. Cheney, J. Bongard, and H. Lipson, "Evolving soft robots in tight spaces," in *Proceedings of the conference on genetic and evolutionary computation*, 2015.

[57] Y. Payan, *Soft tissue biomechanical modeling for computer assisted surgery*. Springer, 2012, vol. 11.

[58] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik, and S. Cotin, "Sofa: A multi-model framework for interactive physical simulation," in *Soft tissue biomechanical modeling for computer assisted surgery*. 2012.

[59] M. Nesme, P. G. Kry, L. Jeřábková, and F. Faure, "Preserving topology and elasticity for embedded deformable models," *ACM Trans. Graph.*, vol. 28, no. 3, 2009.

[60] H. Talbot, S. Marchesseau, C. Duriez, M. Sermesant, S. Cotin, and H. Delingette, "Towards an interactive electromechanical model of the heart," *Interface focus royal society*, 2012.

[61] H. Talbot, F. Roy, and S. Cotin, "Augmented reality for cryoablation procedures," in *SIGGRAPH*, 2015.

[62] I. Peterlik, M. Nouicer, C. Duriez, S. Cotin, and A. Kheddar, "Constraint-based haptic rendering of multirate compliant mechanisms," *IEEE Transactions on Haptics*, vol. 4, no. 3, 2011.

[63] *ArtiSynth*. [Online]. Available: http://artisynth.magic.ubc.ca/artisynth/.

[64] S. Bhavikatti, *Finite element analysis*. New Age International, 2005.

[65] C. Duriez, S. Cotin, J. Lenoir, and P. Neumann, "New approaches to catheter navigation for interventional radiology simulation," *Computer aided surgery*, vol. 11, no. 6, 2006.

[66] A. Theetten, L. Grisoni, C. Duriez, and X. Merlhiot, "Quasi-dynamic splines," in *Proceedings of the ACM Symposium on Solid and Physical Modeling*, 2007.

[67] C. A. Felippa, "A systematic approach to the element-independent corotational dynamics of finite elements," *Center for Aerospace Structures Document Number CU-CAS-00-03, College of Engineering, University of Colorado*, 2000.

[68] N. Kikuchi and J. T. Oden, *Contact problems in elasticity: A study of variational inequalities and finite element methods*. SIAM, 1988.

[69] D. E. Johnson and P. Willemsen, "Six degree-of-freedom haptic rendering of complex polygonal models," in *Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2003. HAPTICS 2003. Proceedings. 11th Symposium on*, 2003.

[70] C. Duriez, F. Dubois, A. Kheddar, and C. Andriot, "Realistic haptic rendering of interacting deformable objects in virtual environments," *Ieee transactions on visualization and computer graphics*, vol. 12, no. 1, 2006.

[71] F. Sha, L. K. Saul, and D. D. Lee, "Multiplicative updates for nonnegative quadratic programming in support vector machines," in *Advances in neural information processing systems*, 2002.

[72] T. Hassan, M. Manti, G. Passetti, N. d'Elia, M. Cianchetti, and C. Laschi, "Design and development of a bio-inspired, under-actuated soft gripper," in *Proceeding of the Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, IEEE, 2015.

[73] A. Fabri and S. Pion, "Cgal: The computational geometry algorithms library," in *Proceedings of the international conference on advances in geographic information systems (ICAGIS)*, 2009.

[74] F. Largilliere, V. Verona, E. Coevoet, M. Sanz-Lopez, J. Dequidt, and C. Duriez, "Real-time control of soft-robots using asynchronous finite element modeling," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[75] Z. Zhang, J. Dequidt, A. Kruszewski, F. Largilliere, and C. Duriez, "Kinematic modeling and observer based control of soft robot using real-time finite element method," in *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[76] M. Thieffry, A. Kruszewski, O. Goury, T. Guerra, and C. Duriez, "Dynamic control of soft robots," in *Preprints of International Conference on Automatic Control (IFAC)*, 2017.